1 The VeriSmall Computer - Tag, Index, and Offset

Counting from the right of our address 0011010100, the three lowest order bits make up the offset, the next two make up the index, and the rest make up the tag. Why, you ask? Well...

- Our cache line size is 8 bytes, and so there are 3 bits in the offset.
- Because our cache is 2-way set associative with a cache line size of 8 Bytes, we know that each “set” in the cache holds 16 Bytes. The total cache size is 64 Bytes, which must mean that there are 4 sets. Therefore, there are 2 bits for the index.
- The tag is comprised of the remaining bits.

2 The VeriSmall Computer - EXECUTION

Unless marked as a HIT, assume it is a cache miss. Okay, here’s what happens:

- Bytes 208-215 are brought into the first block of set 10
- Bytes 488-495 are brought into the first block of set 01
- Bytes 528-535 are brought into the second block of set 10
- Bytes 208-211 HIT in the first block of set 10
- Bytes 16-23 are brought into the second block of set 10, kicking out Bytes 528-535
- Bytes 492-495 HIT in the first block of set 01

The final state of the cache is the first block of set 01 contains Bytes 488-495, the first block of set 10 contains Bytes 208-215, and the second block of set 10 contains Bytes 16-23.
3 You Want Me To Look Where?

The proper searching order is:

- TLB
- Page Table Register
- Page Table
- Data Cache
- DRAM

Theoretically, the access to the TLB and the Page Table Register could be swapped, or even done in parallel. So, you start with a virtual address. The reason for checking the TLB is to see whether or not the translation of your virtual address to a physical address is there, which would save you a trip to main memory where the Page Table resides. Unfortunately, it is not there. So, you check the Page Table Register which is just a special register that points to the current Page Table in memory. It points you to the Page Table, and you get your address translation from there. Now that you have your physical address, you can check the physically addressed Data Cache to see if you can avoid going to main memory for your data. Unfortunately, it is not there, either. So, you are forced to go to DRAM one more time to get the data you are looking for.

4 A Simple Virtual Memory System

(a) The cache is physically addressed since it is the physical page number that is used to index into it.
(b) The TLB is not direct mapped for a couple of reasons. First of all, it is clear that there are no index bits in the virtual address. This immediately tells you that the virtual addresses are all mapping to a single set, and that the TLB is in fact fully-associative. Another clue is the set of comparators that are doing a parallel comparison of the virtual page number. There would be no need for this in a direct mapped cache since there is only one tag to compare once you’ve indexed into a direct mapped cache.
(c) There is not enough information to determine if the cache is write-allocate or write-around. All of the hardware pictured is simply the basic stuff needed to query the cache and there is nothing to show what the cache’s policy might be on a store miss.
(d) There are $2^{12}$ bytes or 4 KB in a page. You know this because there are 12 Page offset bits. Similar to the way the number of Byte offset bits determines your cache line size, the number of Page offset bits determines your page size.
(e) The Cache holds $2^{16}$ bytes or 64 KB of data. You know this because since the Cache index is 14 bits, there must be $2^{14}$ sets in the cache. Since the Cache is clearly direct-mapped and your Byte offset is 2 bits long, it must be that the cache line size is 4 Bytes, and that each set holds $2^2$ Bytes. $2^{14} \times 2^2 = 2^{16}$. 
5 Bus Basics

(a) Lowest latency? PCI bus. The other buses will typically connect you to something external to your computer.

(b) The program notifying the OS that it wants to do an I/O operation is an internal interrupt. An example of an external interrupt is some I/O device doing the signalling, as opposed to the program.

(c) You could do a few different things to increase bandwidth: increase the number of lines, have separate lines for address and data, or increase the clock rate.

6 Daisy, Daisy, I’m Half Crazy...

The grant line should go from the Bus Arbiter’s ”grant out” line to the High Priority Device through its ”grant in” line, and out its ”grant out” line into the Low Priority Device’s ”grant in” line. Both devices’ request lines should go into the Bus Arbiter’s request line, and both devices’ release lines should go into the Bus Arbiter’s release line.

7 Please Do Not Leave Your Baggage Unattended

(a) For each image processed, there must be 4 MB of data transferred on the system bus. First, each image which is 1 MB in size must be transferred from the Ethernet Controller directly to Memory. Secondly, the Processor needs to bring it across the system bus to do whatever work it needs to do. Third, since the cache has twice as many conflict or capacity misses as it does compulsory misses for each image, the image will have to be fetched piecemeal back into cache a total of 2 additional times. Thus, the image travels along the system bus a total of 4 times, resulting in 4 MB per image.

(b) In order to find the maximum throughput of the system, we have to find the bottleneck. We know the system bus can handle 400 MB/sec, which essentially boils down to 100 images/sec. given the results from part (a). However, the Ethernet bandwidth is only 12.5 MB/sec. Since there are two Fast Ethernets, a total of 25 MB can be transferred to the Ethernet controller per second. Clearly the system can only handle 25 images/sec.

(c) If we upgrade to 125 MB/sec for each Ethernet, then the two combined could deliver 250 MB/sec to the Ethernet Controller. This would support 250 images/sec. However, it is the system bus that is now the bottleneck, only being able to handle 100 images/sec as seen in part (b).

8 Farewell...

Good luck on the final, everyone! It has been a blast being your TA. Hope to see some of you in CSE 105 next quarter!

-Greg